

## CountDownLatchTest1.java

```
1 import java.util.ArrayList;
9 /**
10 *
11 * 通过CountDownLatch和CyclicBarrier实现运动员赛跑准备->发令枪开跑->到达终点计算成绩
12 *
13 * @author renzenggang
14 *
15 */
16 public class CountDownLatchTest1 {
17
18     //运动员数量
19     private static int SPORTSMAN_COUNT = 10;
20
21     private static final Random random = new Random();
22
23     // 用于判断发令之前运动员是否已经进入准备状态，需要等待10个运动员准备就绪，占有锁，等待10个运动员完成，释放锁。
24     private static CountDownLatch readyLatch = new CountDownLatch(SPORTSMAN_COUNT);
25     // 用于判断裁判是否已经发令，占有锁，等待裁判发令完成，释放锁
26     private static CountDownLatch startLatch = new CountDownLatch(1);
27     // 设置终点屏障，用于计算成绩
28     private static CyclicBarrier cb = new CyclicBarrier(SPORTSMAN_COUNT, new
    Runnable() {
29
```

## CountDownLatchTest1.java

```
30     @Override
31     public void run() {
32
33         CountdownLatchTest1.transcript
34             .sort((Sportsman p1, Sportsman p2) -> p1.getTranscript() -
35             p2.getTranscript());
36
37         System.out.println("排名成绩单: " + CountdownLatchTest1.transcript);
38
39         CountdownLatchTest1.transcript.clear();
40     }
41 });
42 // 成绩单
43 private static List<Sportsman> transcript = new
44     ArrayList<Sportsman>(SPORTSMAN_COUNT);
45
46 public static void main(String[] args) {
47     // 用于判断发令之前运动员是否已经进入准备状态, 需要等待10个运动员准备就绪, 占有锁, 等待10
48     // 个运动员完成, 释放锁。
49     // CountdownLatch readyLatch = new CountdownLatch(SPORTSMAN_COUNT);
50     // 用于判断裁判是否已经发令, 占有锁, 等待裁判发令完成, 释放锁
51     // CountdownLatch startLatch = new CountdownLatch(1);
```

## CountDownLatchTest1.java

```
51
52     // 启动10个线程，也就是10个运动员，做准备工作
53     for (int i = 0; i < SPORTSMAN_COUNT; i++) {
54         Thread t = new Thread(new RunTask((i + 1) + "号运动员", readyLatch,
55             startLatch));
56         t.start();
57     }
58     // 当前运动员在其他运动员准备就绪前一直等待，也就是说等readyLatch倒数计数器为0之前一直等
59     待
60     try {
61         readyLatch.await();
62     } catch (InterruptedException e) {
63         e.printStackTrace();
64     }
65     // 裁判发令，释放锁
66     startLatch.countDown();
67     System.out.println("裁判：所有运动员准备完毕，开始跑...");
68
69 }
70 // 运动员
71 static class Sportsman {
72     private String name;
73     private int transcript;
```

## CountDownLatchTest1.java

```
73
74     public Sportsman(String name, int transcript) {
75         this.name = name;
76         this.transcript = transcript;
77     }
78
79     @Override
80     public boolean equals(Object obj) {
81         boolean result = false;
82         if (obj instanceof Sportsman) {
83             result = ((Sportsman) obj).getTranscript() == this.transcript;
84         }
85         return result;
86     }
87
88     @Override
89     public String toString() {
90         return this.name + ":" + this.transcript + " ms";
91     }
92
93     public String getName() {
94         return name;
95     }
96
97     public int getTranscript() {
```

## CountDownLatchTest1.java

```
98         return transcript;
99     }
100
101 }
102
103 // 跑任务
104 static class RunTask implements Runnable {
105
106     private Lock lock = new ReentrantLock();
107
108     private CountDownLatch ready;
109     private CountDownLatch start;
110     private String name;
111
112     /**
113      *
114      * (构造方法)
115      *
116      * @param ready
117      * @param start
118      * @param name 运动员名称
119      */
120     public RunTask(String name, CountDownLatch ready, CountDownLatch start) {
121         this.ready = ready;
```

## CountDownLatchTest1.java

```
122         this.start = start;
123         this.name = name;
124     }
125
126     @Override
127     public void run() {
128         lock.lock();
129         try {
130
131             // 1. 写运动员准备就绪的逻辑,准备readyTime秒
132             int readyTime = random.nextInt(1000);
133             System.out.println(name + ":我需要" + readyTime + "秒的时间准备。");
134             try {
135                 Thread.sleep(readyTime);
136             } catch (InterruptedException e) {
137                 e.printStackTrace();
138             }
139             System.out.println(name + "我已经准备完毕! ");
140             // 释放锁readyLatch-1, 表示一个运动员已经就绪
141             ready.countDown();
142             try {
143                 // 等待裁判发开始命令
144                 start.await();
145             } catch (InterruptedException e) {
```

# CountDownLatchTest1.java

```
146
147     }
148     System.out.println(name + ": 开跑...");
149     int costTime = random.nextInt(500);
150     try {
151         Thread.sleep(costTime);
152     } catch (InterruptedException e) {
153         e.printStackTrace();
154     }
155     System.out.println(name + ": 开跑到达终点。成绩:" + costTime + "ms");
156     transcript.add(new Sportsman(name, costTime));
157     // 等待成绩
158     cb.await();
159 } catch (Exception e) {
160
161 } finally {
162     lock.unlock();
163 }
164
165     }
166
167 }
168
169 }
```